JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2005**

Vol. 4, No. 9, November-December 2005

The JBoss Integration Plug-in for IntelliJ IDEA

Douglas Lyon, Fairfield University, Fairfield CT, U.S.A.
Martin Fuhrer, President of Furher Engineering AG, Biel, Switzerland
Thomas Rowland, Pitney Bowes, Shelton CT, U.S.A.

Take three months to prepare
your machines and three months
to complete your siege engineering.
– Sun Tzu

Abstract

This paper is the final in a series of papers that describe a new plug-in for enabling the integration of the IntelliJ IDEA IDE with the JBoss application server. The JBoss plug-in was first conceived and implemented by Martin Fuhrer at Fuhrer Engineering.

Parts 1, 2 and 3 discussed how to download and install the plug-in, how to create a project containing a web module, a session bean, a servlet, and an application module for deployment to JBoss.

This paper discusses deployment and execution, showing how to create a remote deployment method using SSH Transfer, and a run configuration for both local and remote deployment.

1 PREPARING FOR DEPLOYMENT

Without an automated environment, deployment is often a painful process resulting in developers spending a great deal of time performing application deployment tasks. Most organizations strive for better management of this critical phase in the development cycle. By automating these tasks, a team can execute deployment in less time with more repeatable, predictable and measurable results. The IntelliJ IDEA provides built-in deployment support for J2EE applications that we will use via the JBoss plug-in. This integrated deployment environment will allow us to connect to the JBoss application server and copy the application module (the EAR file) to it automatically.

For remote deployment, before deploying the application to the JBoss server, you have to create a deployment method. Local deployment is the simplest and does not require any additional setup, but in most environments, the application server resides on a remote

Cite this column as follows: D. Lyon, M. Fuhrer and T. Rowland: "The JBoss Integration Plug-in for IntelliJ IDEA", in *Journal of Object Technology*, vol. 4, no. 9, November-December 2005, pp.11-21, http://www.jot.fm/issues/issue_2005_11/column2

machine. There are several remote deployment methods available. The following deployment methods are available in IntelliJ IDEA via the JBoss plug-in:

Local

This deployment method is used when the JBoss server is running on the developer's machine. No additional setup is required.

SSH Transfer

Provides a secure connection between the local client and a remote host. To make the application accessible by the JBoss server, the EAR file is transferred to the remote machine using an SSH (Secure Shell Client) connection prior to deployment. The SSH connection is using SSH2 with either password or public key authentication. You have to specify a temporary directory on the remote machine where the EAR file is temporarily stored, e.g. /tmp.

FTP Transfer

This works exactly like the SSH transfer deployment method, but an FTP connection is used to transfer the EAR file instead of an SSH connection. The FTP connection may be either active or passive.

Shared Filesystem

This may be used when both the local developer's machine and the remote deployment machine have a common filesystem, e.g. through NFS, Samba, or whatever. You have to specify a local path and a remote path, both pointing to the same shared directory.

HTTP Callback

With HTTP callback the EAR file isn't transferred to the remote machine, but the JBoss server is told to open an HTTP connection back to the developer's machine and load the EAR file from there. This implies that the developer's machine has to be visible by the remote machine, i.e. the developer's machine may not be connected to the network through DHCP.

To create a new SSH Transfer deployment method select the F*ile:Settings* menu item to bring up the *Settings* dialog box. Select the *JBoss Plugin* control device, as shown in Figure 1.1.

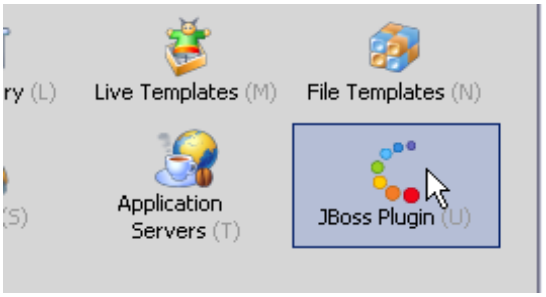


Figure 1.1 The JBoss Plugin Control Device

Select *SSH Transer* from the *Add* button's dropdown in the *JBoss Plugin* dialog box, as shown in Figure 1.2

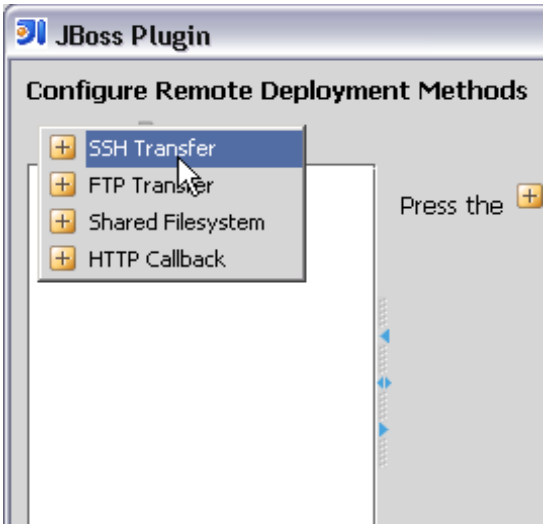


Figure 1.2 Creating an SSH Transfer remote deployment method in the JBoss Plugin dialog

Enter the connection data required to connect to the remote machine and test the connection by selecting *Test Connection*, as shown in Figure 1.3.

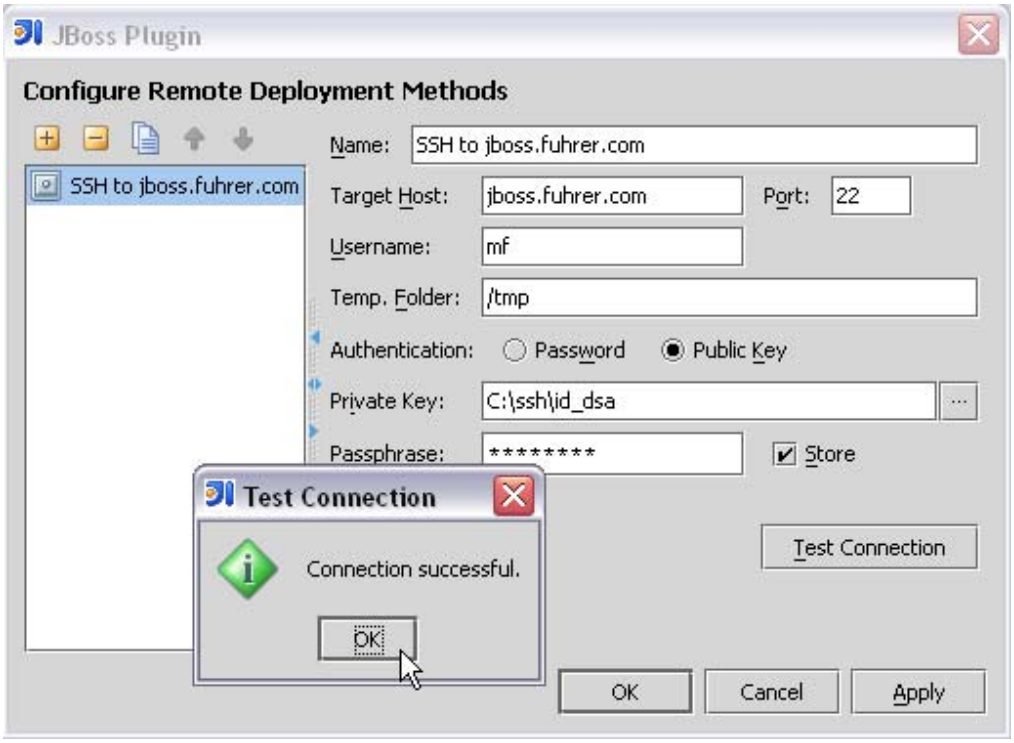


Figure 1.3 Testing the SSH Transfer connection

After the connection has succeeded, select *OK*.

You are now ready to create a run configuration, as shown in the following section.

2 CREATING A RUN CONFIGURATION

The second part to preparing deployment is the setup of a run configuration. Here, you specify application server, startup options, and the modules that are to be deployed. You can create as many run configurations as you like. In this section we will create a *local* run configuration for a local deployment, and a *remote* run configuration using the SSH Transfer deployment method we just created.

To access the run configuration setup, select the *Edit Configurations* menu item from the *Run* menu, as shown in Figure 2.1.

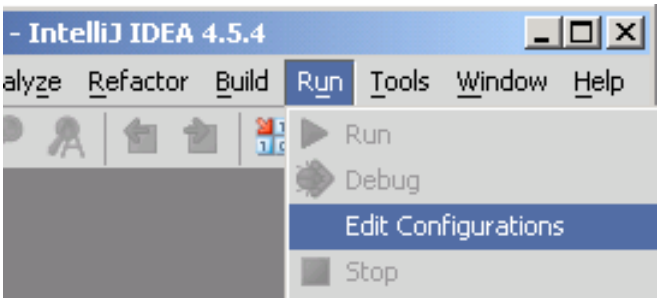


Figure 2.1 The Edit Configurations menu item

Select the *JBoss Server* tab and then select *Local* from the *Add* button's dropdown, as shown in Figure 2.2

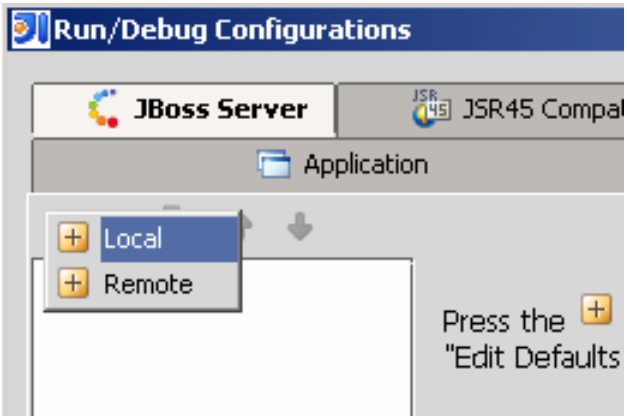


Figure 2.2 Creating a new Local run configuration

Enter a name for the new configuration. On the *Server* panel, select the JBoss application server and select the *default* server instance. Checking the *Start Browser* checkbox will launch the browser when you launch the application, so you don't need to have the browser open at the time.

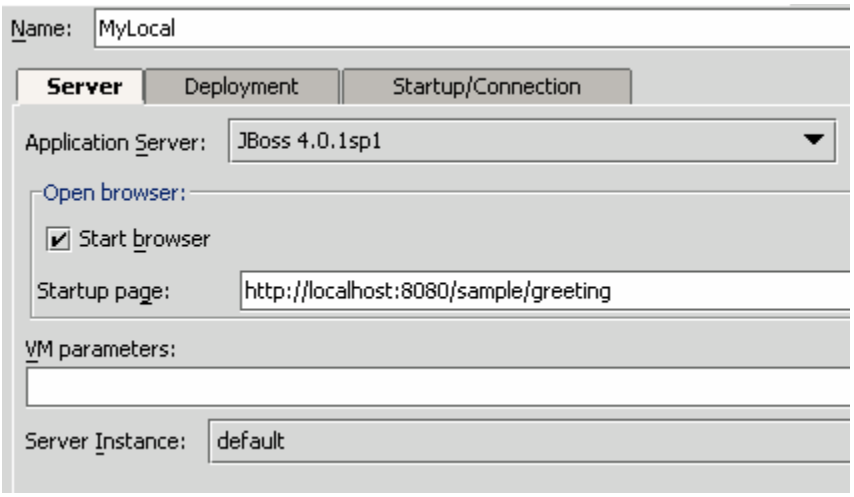


Figure 2.3 Edit Configurations – local deployment Server panel

On the *Deployment* panel, you specify the modules that are to be deployed. Select the *app* module and then select *app.ear* from the dropdown. Remember *app.ear* is our EAR file containing our entire enterprise application that we want to deploy.

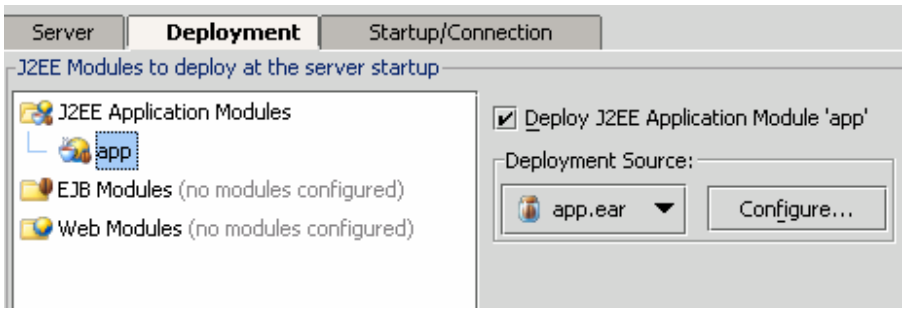


Figure 2.4 Edit Configurations – Deployment panel

The *Startup/Connection* panel allows you to configure startup and shutdown scripts and parameters to be passed to the JBoss server. The defaults as shown in figure 2.5 should be acceptable.

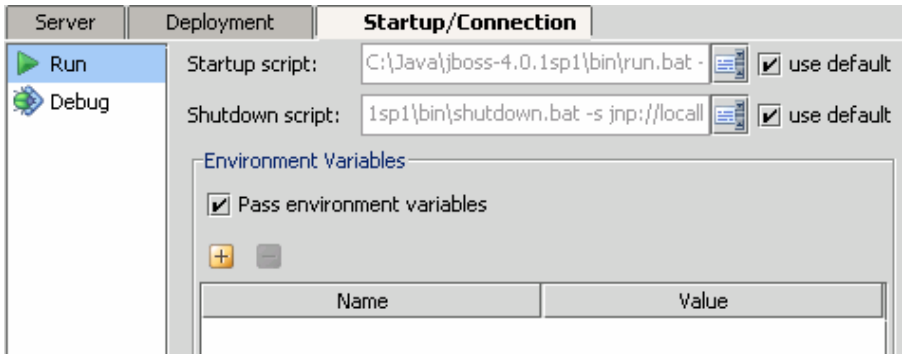


Figure 2.5 Edit Configurations – Startup/Connection panel

Select *OK* and your local run configuration is complete. When you run your application the EAR file will be deployed, JBoss will be started and a new browser instance will be launched.

For remote deployment, select *Remote* from the *Add* button's dropdown in the *JBoss Server* tab, as shown in Figure 2.6

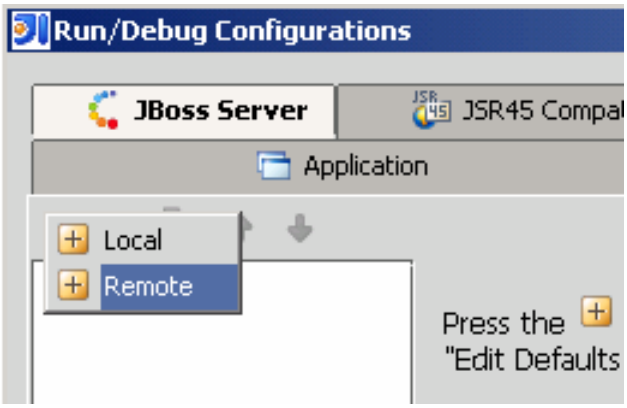


Figure 2.6 Creating a new Remote run configuration

Enter the host name of your remote host running the application server in *Remote Connection Settings*. The name must match the host name you entered in the remote deployment method. As soon as you entered the correct host name the list of available deployment methods is populated by all matching methods. Choose the deployment method you want to use to deploy the EAR file to the application server. The complete setup for the remote configuration is shown in Figure 2.7.

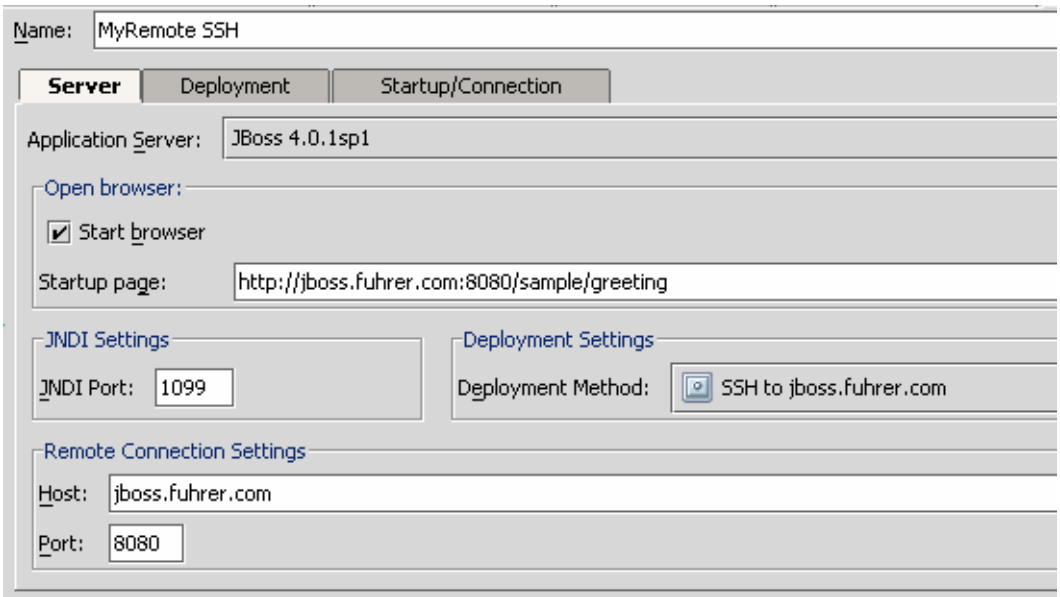
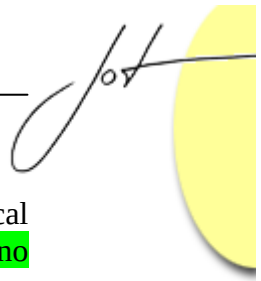


Figure 2.7 Edit Configurations – remote deployment Server panel



Select the *Deployment* tab and use the settings as shown in Figure 2.4 for the local deployment method, as they will be the same. The *Startup/Connection* panel contains no settings for a remote run configuration.

You have now finished creating a run configuration and are ready to run and deploy your application.

3 RUNNING AND DEPLOYING THE APPLICATION (WHAT COULD GO WRONG?)

This section discusses deployment and execution. We will use the remote configuration in our examples.

Before the application can be deployed, you should make sure that JBoss is up and running if you are using remote deployment. Typically, you will invoke a run script in the JBoss bin directory (run.bat for windows, run.sh for Unix, etc). For example:

```
sh run.sh
```

If you are using the local deployment method created above, JBoss will automatically be started. In both cases, a new browser session will be launched.

After a few messages are emitted from the console, you should see a message that looks something like:

```
05:39:01,664 INFO  [Server] JBoss (MX MicroKernel) [4.0.1RC1
        (build: CVSTag=JBoss_4_0_1_RC1 date=200411041143)]
        Started in 1m:16s:917ms
```

It is a good sign if there are no exceptions during startup. If there are, you may have a configuration problem (e.g., 4.0.1RC1 is not compatable with JDK1.5, etc.). To further confirm the correct operation of JBoss, you should visit the application server at the default port of 8080.

If JBoss is running, you should be able to see the JBoss console, as shown in Figure 3.1.

Figure 3.1. The JBoss Console

Select *Run* from the IntelliJ IDE and, if all goes well, you should see some messages like the ones shown in Figure 3.2.

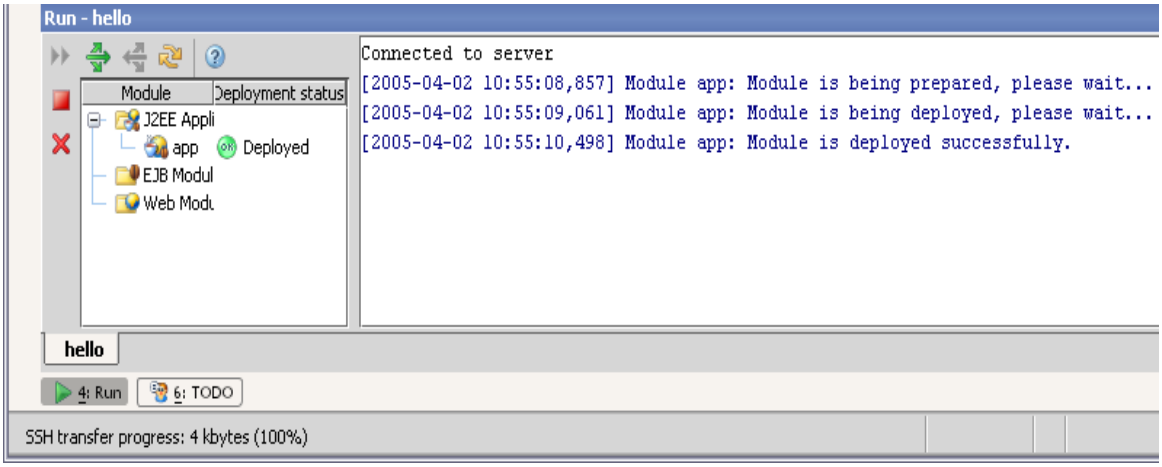


Figure 3.2 Successful Deployment

Before the application is deployed into the JBoss server, the EAR file is transferred by SSH into the temporary folder configured in the deployment method, e.g. /tmp. The EAR file is then picked up by the JBoss server from there. Any exceptions or errors occurring during deployment are logged in the run window of the IntelliJ IDE.

One common problem is that the web browser is started by the IntelliJ IDE before the deployment of the application has completely finished. This results in a missing

resource error displayed in the browser as shown in Figure 3.3. Reload the web page manually after deployment has finished to display the servlet correctly.

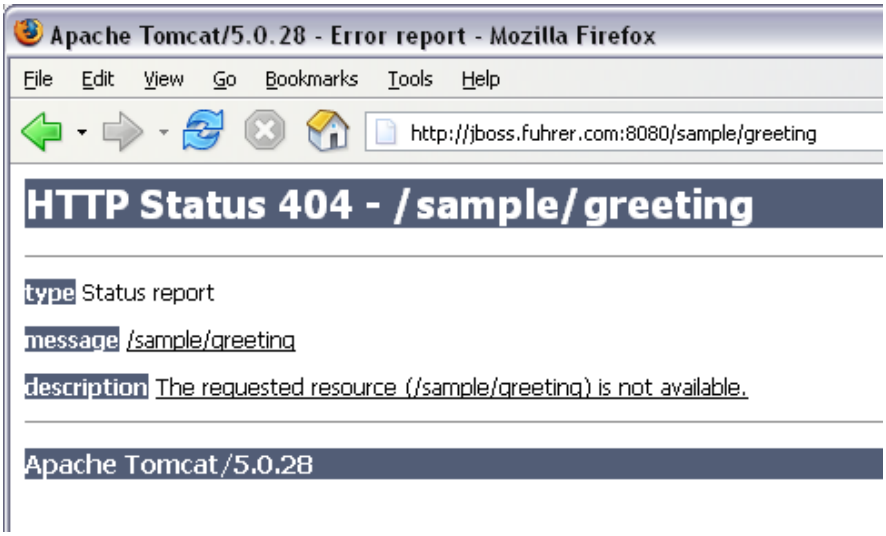


Figure 3.3 Missing Resource Error

If the web server is running properly, the application is deployed, but the servlet is still not displaying in the browser, you must start to examine log files. There are log files on the development machine and log files on the application server.

To examine the development machine's log files, look at:

```
<userHome>/.IntelliJIdea/system/log
```

To examine the application server's log files, look at:

```
<jbossHome>/server/default/log
```

Sometimes an error will occur and the exception will be emitted by the servlet. One common error is a JNDI lookup failure that results in a message like:

```
"ejb not bound" or less likely "Could not dereference object".
```

Check for any JNDI name mismatches in the EJB and/or servlet configuration to solve the problem. Make sure you are using the correct ejb logical reference name in your servlet code.

Check that jboss.xml (the JBoss-specific EJB deployment descriptor) has the correct mapping of the ejb name to the JNDI name:

```
<jboss>
    <enterprise-beans>
        <session>
            <ejb-name>HelloEJB</ejb-name>
            <local-jndi-name>hello</local-jndi-name>
        </session>
    </enterprise-beans>
</jboss>
```

Check that jboss-web.xml (the JBoss-specific web deployment descriptor) has the correct mapping of the ejb logical reference to the JNDI name:

```
<jboss-web>
    <ejb-local-ref>
        <ejb-ref-name>ejb/hello</ejb-ref-name>
        <local-jndi-name>hello</local-jndi-name>
    </ejb-local-ref>
</jboss-web>
```

If all goes well, you should see a reply, like that of Figure 3.4.

Figure 3.4 The Sample Greeting Applet

4 CONCLUSION

The JBoss plug-in is freely available and its download is integrated into the IntelliJ IDEA IDE. The process of creating modules and entering in data is error-prone and tedious. Ideally, there should be an easier way to incorporate EJB's into the development environment.

A common source of fragility is the JNDI lookup:

```
HelloHome home = (HelloHome) new
InitialContext().lookup(
        "java:comp/env/ejb/hello");
```

If the mapping, at run-time, should fail, an *ejb not bound* exception will be thrown by the servlet. It would probably be better software engineering if such an error were a syntax error and not a run-time error. This would probably mean having to engineer JNDI out of the process, an activity that is beyond the scope of this paper.

About the authors

After receiving his Ph.D. from Rensselaer Polytechnic Institute, **Dr. Lyon** worked at AT&T Bell Laboratories. He has also worked for the Jet Propulsion Laboratory at the California Institute of Technology. He is currently the Chairman of the Computer Engineering Department at Fairfield University, a senior member of the IEEE and President of DocJava, Inc., a consulting firm in Connecticut. E-mail Dr. Lyon at Lyon@DocJava.com. His website is http://www.DocJava.com.

Martin Fuhrer has a degree as engineer in computer science from the School of Engineering and Information Technology in Biel/Switzerland. He is founder and president of Fuhrer Engineering Inc., a software development company located in Biel/Switzerland. He's mainly working in the field of web-based financial services and the online processing of realtime stock exchange data. He can be reached at info@fuhrer.com or through http://www.fuhrer.com.

Thomas Rowland has a B.S. in Electrical Engineering and an M.S. in Software Engineering. He has been consulting as a Software Engineer for the past four years, working for Pfizer Pharmaceutical, Travelers Life & Annuity, and currently at Pitney Bowes. He has also worked for Hyperion Solutions for over 5 years. Mr. Rowland has also had some teaching stints along the way. He is listed in the National Register's 2005-2006 edition of the Who's Who in Executives and Professionals. He resides in Connecticut and can be reached at rowlandtf@netscape.net.